REMARKS

Claims 1 through 21 are in prosecution. Claims 1 to 6 are rejected as non-statutory matter under 35 USC 101. Claims 1 to 6 and 15 to 20 are rejected as anticipated under 35 USC 102(b) by Cascio et al (US Pub 2002/00918180, hereinafter "Cascio". Claims 8 to 14 are rejected under 35 USC 103 as obvious over Cascio in view of DaCosta et al (US Pub. 2002/0120725), hereinafter "DaCosta".

The Section 101 Rejection

Examiner maintains the Section 101 rejection, stating that applicants intend that the language of claims 1-7, directed to a computer storage medium, encompass signals, presumably as in transmitted signals. Examiner suggests that claim 1 be amended such that "contain" in line 1 of claim 1 be replaced with a variant of "store". Attorney has examined the specification and can find no indication one way or another of how applicants intend the claim language to be interpreted. However, claim 1 is amended as suggested without prejudice to advance this prosecution. Examiner is respectfully requested to withdraw this Section 101 rejection in view of this amendment.

The Section 102 Rejection

Applicants recognize that claim 1 does not adequately capture the high level difference between the present invention and Casico. Claim 1 is amended to more distinctly define the invention. In addition, a number of the dependent claims are also amended to place them in better form for allowance or appeal. Applicants believe that the remaining independent claims are in good condition. More about these amendments below.

In response to applicant's arguments in the preceding Office Action response, Examiner states:

A1. Applicant argues that Cascio does not teach a plurality of agents for performing functionality of the system.

RI. Examiner does not agree. The Applicant ... states that "...a procedural agent would execute as a single process...", hence an agent is a process

Applicants: Agree that an agent is a process.

. With that in mind the Examiner likes to note paragraphs 53, 60, 73, 75 which describe different processes for performing different functionality much like an individual agent to perform certain functionality as explained by the Applicant.

Applicants: Paragraphs 53 and 60, begin a description of creating a data pattern rule. This offers no information on single vs multiple processes.

Paragraph 73 describes the single process of matching all rules to the incoming screen data. The text references Fig. 6, which is the single process.

Paragraph 75 discusses the output document that is produced by the single process.

Contrary to Examiner's statement, these paragraphs do not describe different processes for matching rules and extracting screen data. Only at the single process flowchart of Fig. 6 is where rules are matched and data extracted.

Then the Examiner would like to know paragraphs 41, 44-45 in which Cascio provides direct discloser of the system capable of being ran on multiple diverse computer environments including multi-core and multi-processor computing

environments, thus Cascio gives another scenario of how Cascio has multiple agents performing desired functionality from the system of Cascio.

Applicants: This argument has been addressed by applicants in the response to the preceding office Action. Multiple processors do not mean that each processor is executing a different process, or rule in this case. Each processor might be running the same single process on different areas of the screen data, to improve performance. The present invention might also be spread over multiple processors by, for example, running a separate and different agent on each processor. There is a significant architectural difference between splitting a single process across different data and running different processes on different processors.

Then again Cascio also provides further evidence of a plurality of agents by explaining the system can be ran in a multi-tiered architecture, which know in the art has a plurality of agents to invoke functionality from the Cascio system. (Examiner cites Wikipedia for a discussion of multi-tier processing)

Applicants: Wikipedia actually bolsters applicant's arguments. At http://en.wikipedia.org/wiki/Multitier_architecture, Wikipedia says:

In software engineering, multi-tier architecture (often referred to as n-tier architecture) is a client-server architecture in which an application is executed by more than one distinct software agent. For example, an application that uses middleware to service data requests

between a user and a database employs multi-tier architecture. The most widespread use of "multi-tier architecture" refers to three-tier architecture.

Notice that Wikipedia says that "an application" (typically a single process) is executed by more than one software agent. This is the equivalent of using multiprocessors, just that processors are replaced with software agents. This does not, however, mean that each software agent is executing a different process.

Examiner continues:

A2. Applicant argues Cascio does not teach determining whether tow or more match regions overlap.

R2. Examiner does not agree, in paragraph 72 it is directly explained that the system detects same matched areas and does not keep the same areas with the end goal being the satisfy the rule to have a complete filed (all regions collected) such that extraction can then take place. It is explained that the same regions that are matched are not needed and there for were detected. Cascio may not use the term "overlap" but the function of overlapping is present to at least paragraph 72.

Applicants: Applicants do not completely understand Examiner's argument above. But a simple reading of paragraph 72 clearly shows that there is not even a suggestion of detecting overlapped portions of screen data. Paragraph 72 merely repeats the blocks steps of Fig. 6, which does not address this issue in the least.

Cascio and the present application solve the same problem, that of transforming legacy character-based terminal screens into a form suitable for graphical user interface (GUI) applications, without requiring end-user programming. The methods used by Cascio and the present invention are fundamentally different and it is argued that the present claims sufficiently reflect the fundamental difference of this application over Cascio. Cascio describes an algorithm in which the designers specify a set of rules, using regular expressions in one embodiment, that allow a computer process to recognize a portion of a screen by matching a rule to the screen data. At the end of the process, all the pre-specified output formats associated with the matching rules are used to build a transformed GUI screen.

Thus, Cascio uses a single process to match rules to the screen data. Cascio specifies in paragraph 47 that more that one processor may simultaneously process rules against the incoming character stream. But, each computer is still executing the same single process and a set of rules to identify a portion of a screen that matches a rule. The use of multiple computers just speeds-up the process.

In contrast, the claimed invention uses a plurality of processes, called agents, in a computer to recognize screen patterns. Unlike Cascio, each agent is conditioned to match a single screen pattern. A Cascio rule is a grouping of words that must be parsed and interpreted by the single Cascio process. The single process must sequence through every rule and compare every rule to the entire contents of a screen of data. On the other hand, each of the agents in this application merely looks for a single data pattern and/or control pattern in the screen data that are unique to the agent. This is a much simpler method of transforming a character screen into a GUI screen that the known art such as Cascio.

Claim 1, is amended to more distinctly claim the invention and distinguish over Casico. For example, the following claim 1 language

code agent for determining which host component types exist in the character based user interface, each agent determining the existence of a different host component type from the other agents;

is amended to the following:

code in each agent for determining ~~which host component types exist in the character based user interface, each agent determining~~ the existence of a different host component type ~~from the other agents~~ unique to the agent;

to capture clearly that, in the present invention, multiple agents each look for a single, different data pattern that the other agents.

This fundamental architectural difference between the present invention and Casico is already clearly present in the remaining independent claims. For example, claim 8 contains the language:

a memory comprising a plurality of agent objects to scan the character based user interface, each agent object determining the existence of a different host component type from the other agents (underling added for emphasis)

The issue of tending to overlapped screen regions is also very important. Examiner alleges that Cascio Fig. 6 and paragraphs 58, 72-73, and 76-77 teach the determination of overlapping regions. Cascio contains no such teaching. The issue of overlapping regions is completely ignored. Fig. 6 is a flowchart of Cascio's single process and contains no step of determining overlapping regions. Paragraph 58 describes using both text and field attributes in a rule, but does not address determining overlapping regions; paragraphs 72-73 describe the single process of Fig. 6 and contains nothing relating to overlapping screen components; paragraph 76 points to Fig. 11 as an example of a GUI transformed version of the character screen of Fig. 7. No overlapping components are shown in either Fig. 7 or 11. Paragraph 77 describes the possibility of transforming a legacy screen into different languages, such as XML, HTML and the like. But, again there is no mention of overlapping screen components.

Similarly, Casico does not deal with prioritizing overlapping regions, as required by a number of dependent claims. In fact, every dependent claim that deals with overlapping regions in any way is completely unanticipated by Cascio, and cannot be made obvious thereby because there is no teaching or recognition of the problem.

The Section 103 Rejection

Claims 8-14 are rejected a being obvious under Cascio in View of DaCosta. Independent claim 8 contains language similar to the multiple agents (processes) discussed above. A minor variation is that the claim refers to a plurality of object agents instead of just agents. It is known to skilled art workers in the field of programming that applications can be implemented as objects (also known as classes) or, alternatively, as procedural code. A procedural agent would execute as a single process as above discussed. An object can contain one or more "methods",

with each method also executing as a single process when called. Therefore, it is argued that claim 8 and its dependent claims also distinguish over Cascio in the use of multiple objects, each conditioned to match a different data pattern.

Dependent claims 2-7 9-14 and 16--21 are allowable because they introduce other distinguishing features that are discussed above, and because of their dependency.

All claims are now believed to distinguish over the art of record and are in condition for allowance. Examiner is respectfully requested to enter this amendment after final and to pass this application to issue.

Respectfully Submitted,

/Jerry Herndon/

Jerry Herndon
Reg. No. 27901

IBM Corporation
Intellectual Property Law
Department T81/Building 503
P.O. Box 12195
Research Triangle Park, NC 27709

(919) 543-0345
FAX: 919-254-4330
Email: jherndon@us.ibm.com